# Java Dünyasında Frameworkler ve Çözdükleri Problemler

1.Spring Framework:

Problemler: Bağımlılık enjeksiyonu (Dependency Injection), IOC (Inversion of Control), aspect-oriented programming (AOP), veritabanı etkileşimi, RESTful servisler, güvenlik gibi birçok alanı kapsar.

Kod Örneği:

```
import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class MyService {

    private final MyRepository repository;

    @Autowired

    public MyService(MyRepository repository) {

        this.repository = repository;

    }

    public void doSomething() {

        // Veritabanı işlemleri veya diğer işlemler...

    }

}

@Repository

public interface MyRepository extends JpaRepository<MyEntity, Long> {

}
```

2. Hibernate:

Problemler: Nesne-ilişkisel eşleme (Object-Relational Mapping - ORM), veritabanı işlemleri, SQL sorgularının otomatik oluşturulması, ilişkisel veritabanı modeli ve nesne modeli arasındaki dönüşümler.

Kod Örneği:

```
@Entity
@Table(name = "products")
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private double price;
}
```

3. Apache Struts:

Problemler: Model-View-Controller (MVC) modeline dayalı web uygulamaları geliştirme.

Kod Örneği:

```
public class HelloWorldAction extends ActionSupport {
    public String execute() {
        setMessage("Merhaba, Dünya!");
        return SUCCESS;
    }
}
```

4. JUnit:

Problemler: Birim testler yazma ve yürütme.

Kod Örneği:

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;
public class MyTest {
    @Test
    public void testAddition() {
        int result = Calculator.add(3, 4);
```

```
assertEquals(7, result);
    }
}
```

5. Apache Kafka:

Problemler: Yüksek ölçekli, dağıtık, iş parçacığı olmayan ve dayanıklı mesaj kuyruğu sağlama.

Kod Örneği:

```
Properties props = new Properties();
props.put("bootstrap.servers", "localhost:9092");
props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
Producer<String, String> producer = new KafkaProducer<>(props);
producer.send(new ProducerRecord<>("my-topic", "key", "value"));
```

# Soa Nedir ?

SOA (Service-Oriented Architecture), Java ile yazılmış yazılımların tasarımında kullanılan bir mimari yaklaşımıdır. SOA, yazılımların işlevselliğini hizmetlere ayırır ve bu hizmetlerin birbirleriyle iletişim kurabilmelerini sağlar. Her bir hizmet, genellikle belirli bir işlevselliği temsil eder ve diğer hizmetlerle standart arayüzler üzerinden iletişim kurar.

Java, SOA için oldukça uygun bir dil ve platformdur. Java, nesne yönelimli programlama (OOP) yeteneklerine sahip olduğu için hizmetler arasındaki bağımlılıkları yönetmek ve veri yapılarını tanımlamak için idealdir. Ayrıca Java'nın geniş kütüphane desteği ve güçlü web servisleri desteği (SOAP ve REST gibi) SOA uygulamaları geliştirmek için avantaj sağlar.

SOA kullanılarak geliştirilen Java tabanlı uygulamalar, genellikle esneklik, ölçeklenebilirlik ve tekrar kullanılabilirlik gibi faydalar sağlar. Bu yaklaşım, büyük ölçekli sistemlerin karmaşıklığını yönetmek için yaygın olarak tercih edilir.

Soa prensipleri:

**Interoperability**

Servislerin diğer servislerle birlikte çalışabilir olduğunu belirtir. Birlikte çalışabilirlik için ortak bir biçim kullanılır.

**Reusability**

Servislerin tekrar kullanılabilir olduğunu belirtir.

**Abstraction**

Servis iç yapısının servis kullanıcıları tarafından gizlenmesidir.

**Facade**

Servis ve servisi kullanan arasındaki bir bileşen/kabuk olduğunu belirtir.

**Autonomy**

Bir servisin diğer servislerden bağımsız olarak çalışabilir olduğunu belirtir.

**Statelessness**

Servislerin durumsuz olduğunu belirtir. Servislerin durum bilgisi servis isteğine göre şekil alacağını ve sürekli aynı olmadığını belirtir.

**Discoverability**

Servislerin keşfedilebilir olduğunu belirtir.

# WebService Nedir ?

Web servisleri, farklı bilgisayarlar arasında iletişim kurmak için kullanılan bir teknoloji veya mimaridir. İnternet üzerinden (genellikle HTTP protokolü kullanılarak) bilgi alışverişi sağlarlar.

Web servisleri, genellikle XML (Extensible Markup Language) veya JSON (JavaScript Object Notation) formatlarında veri alışverişi yaparlar. Bu, farklı platformlar arasında veri alışverişini kolaylaştırır, çünkü XML ve JSON, dil bağımsız veri aktarımı sağlar.

Web servisleri, genellikle üç ana bileşeni içerir:

SOAP (Simple Object Access Protocol) veya REST (Representational State Transfer) protokolü: Veri iletişimini sağlar ve istemci ile sunucu arasındaki mesajlaşmayı tanımlar.

WSDL (Web Services Description Language): Web servisinin işlevselliğini ve nasıl kullanılacağını tanımlayan bir XML tabanlı belgedir.

UDDI (Universal Description, Discovery, and Integration): Web servislerinin keşfedilmesi ve kullanılabilirliği için bir protokol ve kütüphane sağlar.

Web servisleri, farklı platformlar arasında (örneğin, bir mobil uygulama ile bir web sunucusu arasında) veri alışverişini sağlamak için yaygın olarak kullanılır. Örneğin, bir web servisi, bir müşteri bilgilerini almak veya işlemleri gerçekleştirmek gibi bir işlevi sunabilir. Bu, uygulamalar arasında entegrasyonu kolaylaştırır ve veri paylaşımını standartlaştırır.

Örnek : OpenWeatherMap servisi

# RestfulService Nedir?

RESTful servisler, REST (Representational State Transfer) mimarisine uygun olarak tasarlanmış web servisleridir. REST, internet tabanlı uygulamaların standartlarını tanımlayan bir mimari tarzıdır. RESTful servisler, kaynaklarla (resource) çalışır ve bu kaynaklara HTTP protokolü üzerinden erişim sağlar.

RESTful servislerin temel özellikleri şunlardır:

Kaynaklar (Resources): Her kaynak, benzersiz bir URI (Uniform Resource Identifier) ile temsil edilir. Örneğin, bir blog uygulamasında "yazılar" veya "kullanıcılar" gibi kaynaklar olabilir.

HTTP Metodları: CRUD işlemlerini (Create, Read, Update, Delete) gerçekleştirmek için HTTP metotları kullanılır. GET, POST, PUT, DELETE gibi HTTP metotları, kaynaklara erişim ve değişiklik yapmak için kullanılır.

Temel Medya Türleri (Media Types): Veri değişiminde genellikle JSON veya XML gibi standart medya türleri kullanılır. Bu, istemcilerin ve sunucuların birbirleriyle veri alışverişi yaparken uyumlu olmalarını sağlar.

Stateless (Durumsuz): Sunucu, her bir isteği tek başına işler ve istekler arasında herhangi bir durum saklamaz. Bu, ölçeklenebilirlik ve güvenilirlik açısından avantaj sağlar.

HATEOAS (Hypermedia As The Engine Of Application State): RESTful servisler, istemcilere, kaynaklarla ilişkilendirilmiş linklerin (hypermedia) yanı sıra bu linklerin ne tür işlemler gerçekleştirebileceğine dair bilgi sağlar. Bu sayede, istemciler dinamik olarak kaynaklar arasında gezinebilirler.

RESTful servisler, internet üzerinden veri alışverişi yapmak için popüler bir yaklaşımdır ve çeşitli platformlar arasında iletişimi kolaylaştırır. JSON veya XML gibi standart veri formatları kullanılarak, farklı dillerde ve platformlarda geliştirilen uygulamalar arasında uyumlu veri alışverişi sağlanabilir.

# HttpMethods Nedir ?

HTTP (Hypertext Transfer Protocol), web tarayıcıları ve web sunucuları arasındaki iletişimi sağlayan bir protokoldür. HTTP, çeşitli işlemleri gerçekleştirmek için farklı HTTP metotlarını kullanır. Bu metotlar, sunuculara istemciler tarafından gönderilen isteklerin türünü belirtir ve sunucuların bu isteklere nasıl yanıt vereceğini belirler.

İşte yaygın olarak kullanılan bazı HTTP metotları:

GET: Bir kaynağın (örneğin, bir web sayfası veya bir dosya) alınması için kullanılır. GET istekleri yalnızca veri almak için kullanılır ve sunucunun durumunu değiştirmemelidir. Örneğin, bir web tarayıcısı, bir web sayfasını görüntülemek için sunucuya bir GET isteği gönderir.

POST: Sunucuya yeni bir kaynak eklemek veya varolan bir kaynağı güncellemek için kullanılır. POST istekleri genellikle bir form gönderme işlemi için kullanılır. Örneğin, bir kullanıcı bir web formunu doldurduğunda ve gönder düğmesine bastığında, bu form verileri POST isteği ile sunucuya gönderilir.

PUT: Belirli bir URI'deki mevcut kaynağın durumunu tamamen veya kısmen değiştirmek için kullanılır. PUT isteği, sunucunun belirtilen URI'deki kaynağın tümünü, belirtilen verilerle değiştirmesini ister. Örneğin, bir dosyanın yeni bir sürümünü yüklemek için PUT isteği kullanılabilir.

DELETE: Belirtilen URI'deki bir kaynağı silmek için kullanılır. DELETE isteği gönderildiğinde, sunucu belirtilen URI'deki kaynağı siler. Örneğin, bir web sitesinden bir dosyayı veya bir kaydı silmek için DELETE isteği gönderilebilir.

PATCH: Bir kaynağın kısmi olarak değiştirilmesi için kullanılır. PATCH isteği, belirtilen URI'deki kaynağın bir kısmını güncellemek için kullanılır. PUT isteğiyle benzerdir, ancak PUT isteği, kaynağın tamamen değiştirilmesini isterken, PATCH isteği yalnızca belirli alanların güncellenmesini ister.

HEAD: Bir GET isteğinin yanıtıyla aynı yanıtı alır, ancak yanıt gövdesini içermez. HEAD isteği, bir kaynağın meta bilgilerini (örneğin, boyutu veya türü) almak için kullanılır, ancak kaynağın kendisi alınmaz.

Bu HTTP metotları, web uygulamalarının farklı işlevleri gerçekleştirmesi için kullanılır ve RESTful API'lerde kaynakları işlemek için yaygın olarak kullanılır.

Mehmet Emin Meşe